Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 5

Тема Построение и программная реализация алгоритмов численного интегрирования.

Студент Никуленко И.В.

Группа ИУ7-42Б

Оценка (баллы) _____

Преподаватель Градов В.М.

Москва.
2021 г

Цель работы. Получение навыков построения алгоритма вычисления двукратного интеграла с использованием квадратурных формул Гаусса и Симпсона.

1 Исходные данные

Двукратный интеграл при фиксированном значении параметра τ:

$$\varepsilon(\tau) = \frac{4}{\pi} \int_0^{\pi/2} d\varphi \int_0^{\pi/2} [1 - \exp(-\tau \frac{l}{R})] \cos\theta \sin\theta \, d\theta ,$$

где $$\frac{l}{R} = \frac{2\cos\theta}{1 - \sin^2\theta\cos^2\varphi},$$

θ, φ - углы сферических координат.

Применяется метод последовательного интегрирования. По одному направлению использовать формулу Гаусса, а по другому - формулу Симпсона.

2 Код программы

Код программы представлен на листингах 1-2.

Листинг 1. cogs.py

```
from math import sin, cos, exp
import numpy as np


def get_legandre_func(n):
  if n == 0:
    return lambda x: 1
  elif n == 1:
    return lambda x: x
  else:
    return lambda x: (2 * n - 1) / n * x * get_legandre_func(n - 1)(x) - (n - 1) / n * get_legandre_func(n - 2)(x)


def bisection_method(f, a, b):
  eps = np.finfo(float).eps
  if (f(a) * f(b) >= 0):
    return None
     c = a
  while ((b - a) >= eps):
```

```
        c = (a + b) / 2
        if (abs(f(c)) < eps):
            break
                if (f(c) * f(a) > 0):
            a = c
        else:
            b = c
    return c


def find_legandre_roots(n):
    roots = []
    roots_cnt = 0
    max_cnt = n // 2
    h = 1 / (n + 2)
    i = -1

    while (roots_cnt < max_cnt):
        cur_root = bisection_method(get_legandre_func(n), i, i + h)
        if (cur_root and not cur_root in roots):
            roots.append(cur_root)
            roots_cnt += 1

        i += h
        if i > 0:
            i = -1
            h /= 2

    result = roots[:]
    if n % 2 != 0:
        result.append(0)
    result.extend([-x for x in roots])
    result = sorted(result)
    return result


def linalg_solve(A, B):
    n = len(B)

    for i in range(n):
        for j in range(i + 1, n):
            coefficient = -(A[j][i] / A[i][i])
            for k in range(i, n):
                A[j][k] += coefficient * A[i][k]
            B[j] += coefficient * B[i]
    x = [0] * n
    for i in range(n - 1, -1, -1):
        for j in range(n - 1, i, -1):
            B[i] -= x[j] * A[i][j]
        x[i] = B[i] / A[i][i]
    return x


def leggauss(n):
    B = []
    for i in range(n):
        B.append(0 if i % 2 else 2 / (i + 1))
```

```
    roots = find_legandre_roots(n)

    A = []
    for i in range(n):
        temp = []
        for j in range(n):
            temp.append(roots[j] ** i)
        A.append(temp)
    coefficients = linalg_solve(A, B)
    return roots, coefficients


def gauss(func, a, b, n):
    roots, coefficients = leggauss(n)
    l = 0

    m1 = (b + a) / 2
    m2 = (b - a) / 2

    for i in range(n):
        l += coefficients[i] * func(m1 + m2 * roots[i])
    l = l * m2

    return l


def simpson(func, a, b, n):
    h = (b - a) / (n - 1)
    l = 0
    for i in range((n - 1) // 2):
        l += func(a) + 4 * func(a + h) + func(a + 2 * h)
        a += 2 * h
    l = l * h / 3

    return l


def generate_function(tau):
    sub_func = lambda theta, phi: 2 * cos(theta) / (1 - (sin(theta) * cos(phi)) ** 2)
    main_func = lambda theta, phi, tau: (4 / np.pi) * (1 - exp(-tau * sub_func(theta, phi))) * cos(theta) * sin(theta)
    return lambda theta, phi: main_func(theta, phi, tau)


def integrate(func, phi_func, theta_func, phi_limits, thtea_limits, n, m):
    theta_result = lambda phi: theta_func(lambda theta: func(theta, phi), thtea_limits[0], thtea_limits[1], n)
    result = phi_func(theta_result, phi_limits[0], phi_limits[1], m)
    return result
```

Листинг 2. out.py

```
from cogs import gauss, simpson, generate_function, integrate
import numpy as np
import matplotlib.pyplot as plt
```

```
def print_result(coeffs):
    phi_func = gauss
    theta_func = simpson

    x_values = list()
    y_values = list()
    for t in np.arange(0.05, 10, 0.05):
        func = generate_function(t)
        x_values.append(t)
        y_values.append(integrate(func, phi_func, theta_func, (0, np.pi/2), (0, np.pi/2), coeffs[0], coeffs[1]))

    plt.grid(True)
    plt.plot(x_values, y_values)
    plt.xlabel("Tau", size=15)
    plt.ylabel("Epsilon(Tau)", size=15)
    plt.show()
```

3 Результаты работы

1. Описать алгоритм вычисления n корней полинома Лежандра n-ой степени P (x) n при реализации формулы Гаусса.

Метод половинного деления. Полуинтервал [-1;0) разбивается на малые интервалы, в каждом из которых производится попытка нахождения корня. Если после прохода не найдено достаточно корней, малые интервалы делятся пополам и вновь осуществляется проход, пока не найдено n/2 корней.

2. Исследовать влияние количества выбираемых узлов сетки по каждому направлению на точность расчетов.

Зависимость от количества интервалов по θ:

```
Количество интервалов N(φ) = 3
Количество интервалов M(θ) = 3
Метод прохождения по φ - Симпсон
Метод прохождения по θ - Гаусс
Значение τ =  1
Результат = 0.8886948534129958
Количество интервалов N(φ) = 3
Количество интервалов M(θ) = 5
Метод прохождения по φ - Симпсон
Метод прохождения по θ - Гаусс
Значение τ =  1
Результат = 0.8887830243938746
Количество интервалов N(φ) = 3
Количество интервалов M(θ) = 7
Метод прохождения по φ - Симпсон
Метод прохождения по θ - Гаусс
Значение τ =  1
Результат = 0.8887854071985897
```

Зависимость от количества интервалов по φ:

```
Количество интервалов N(φ) = 3
Количество интервалов M(θ) = 3
Метод прохождения по φ - Симпсон
Метод прохождения по θ - Гаусс
Значение τ =  1
Результат = 0.8886948534129958

Количество интервалов N(φ) = 5
Количество интервалов M(θ) = 3
Метод прохождения по φ - Симпсон
Метод прохождения по θ - Гаусс
Значение τ =  1
Результат = 0.8208551739111585

Количество интервалов N(φ) = 7
Количество интервалов M(θ) = 3
Метод прохождения по φ - Симпсон
Метод прохождения по θ - Гаусс
Значение τ =  1
Результат = 0.8152779841066683
```

Зависимость от количества интервалов по θ:

```
Количество интервалов N(φ) = 3
Количество интервалов M(θ) = 3
Метод прохождения по φ - Гаусс
Метод прохождения по θ - Симпсон
Значение τ =  1
Результат = 0.8057719561370534

Количество интервалов N(φ) = 3
Количество интервалов M(θ) = 5
Метод прохождения по φ - Гаусс
Метод прохождения по θ - Симпсон
Значение τ =  1
Результат = 0.8130182148150474

Количество интервалов N(φ) = 3
Количество интервалов M(θ) = 7
Метод прохождения по φ - Гаусс
Метод прохождения по θ - Симпсон
Значение τ =  1
Результат = 0.8126077758469115
```

Зависимость от количества интервалов по φ:

```
Количество интервалов N(φ) = 3
Количество интервалов M(θ) = 3
Метод прохождения по φ - Гаусс
Метод прохождения по θ - Симпсон
Значение τ =  1
Результат = 0.8057719561370534

Количество интервалов N(φ) = 5
Количество интервалов M(θ) = 3
Метод прохождения по φ - Гаусс
Метод прохождения по θ - Симпсон
Значение τ =  1
Результат = 0.8094601626173515

Количество интервалов N(φ) = 7
Количество интервалов M(θ) = 3
Метод прохождения по φ - Гаусс
Метод прохождения по θ - Симпсон
Значение τ =  1
Результат = 0.809433968670951
```

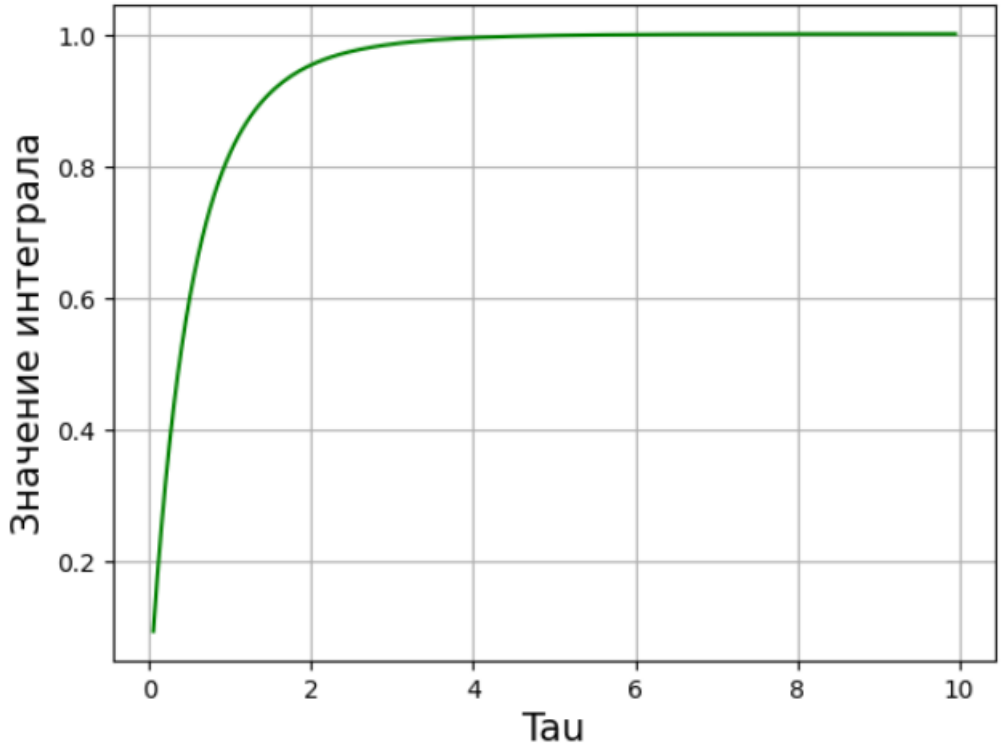
Вывод:

Воспользуемся функцией из библиотеки «SciPy» для вычисления значения интеграла при τ=1

```
from cogs import generate_function
import numpy as np
from scipy.integrate import dblquad

tau = 1
func = generate_function(tau)

answ = dblquad(func, 0, np.pi/2, lambda x: 0, lambda x: np.pi/2)
print("Результат = {:^3}".format(answ[0]))


Результат = 0.8142904188981919
```

Из результатов работы программы видно, что на точность результата в большей степени влияет количество узлов для интегрирования по внешнему направлению. При равенстве количества узлов по внешнему и внутреннему направлению (n = m) метод Гаусса точнее метода Симпсона.

3. Построить график зависимости ε(τ) в диапазоне изменения τ = 0.05-10. Указать при каком количестве узлов получены результаты.




```
Количество интервалов N(φ) = 5
Количество интервалов M(θ) = 5
Метод прохождения по φ - Гаусс
Метод прохождения по θ - Симпсон
Значение τ =  1
Результат = 0.8143810072256769
```

4 Вопросы при защите лабораторной работы

1. В каких ситуациях теоретический порядок квадратурных формул численного интегрирования не достигается.

Если подынтегральная функция не имеет производных до определенного порядка, то теоретический порядок точности не достигается. Например, если на отрезке интегрирования не существуют 3-я и 4-я производные, то порядок точности формулы Симпсона будет только 2-ой.

2. Построить формулу Гаусса численного интегрирования при одном узле.

Корни полинома Лежандра:

$P_1(x) = x$

$x = 0$

$$\begin{cases} \sum_{i=1}^{1} A_i = 2 \\ A_1 = 2 \end{cases}$$

Формула Гаусса:

$$\int_{-1}^{1} f(t)\,dt = A_1 f(x) = 2f(0)$$

Замена переменной:

$$\int_{a}^{b} f(x)\,dx = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b+a}{2} + \frac{b-a}{2} t\right) dt = (b-a) f\left(\frac{b+a}{2}\right)$$

3. Построить формулу Гаусса численного интегрирования при двух узлах.

Корни полинома Лежандра:

$P_2(x) = \frac{1}{2}(3x^2 - 1)$

$x_1 = \sqrt{1/3}; x_1 = -\sqrt{1/3}$

Система для поиска коэффициентов:

$$\begin{cases} \sum_{i=1}^{2} A_i = 2 \\ \sum_{i=1}^{2} A_i x_i = 0 \end{cases}$$

$A_1 = 1$; $A_2 = 1$

Формула Гаусса:

$$\int_{-1}^{1} f(t)\,dt = A_1 f(x_1) + A_2 f(x_2) = f\left(\sqrt{\frac{1}{3}}\right) + f\left(-\sqrt{\frac{1}{3}}\right)$$

Замена переменной:

$$\int_{a}^{b} f(x)\,dx = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b+a}{2} + \frac{b-a}{2} t\right) dt = \frac{b-a}{2} \left(f\left(\frac{a(\sqrt{3}+1) + b(\sqrt{3}-1)}{2\sqrt{3}}\right) + f\left(\frac{a(\sqrt{3}-1) + b(\sqrt{3}+1)}{2\sqrt{3}}\right)\right)$$

4. Получить обобщенную кубатурную формулу, аналогичную (6.6) из лекции №6, для вычисления двойного интеграла методом последовательного интегрирования на основе формулы трапеций с тремя узлами по каждому направлению.

N = M = 2

Формула трапеций: $\int_a^b f(x)dx = h\left(\frac{f(x_0)}{2} + f(x_1) + \frac{f(x_2)}{2}\right)$

Обобщенная формула:

$$\int_c^d \int_a^b f(x,y)dx\,dy = \int_a^b F(x)dx = \frac{h_x}{2}(F_0 + 2F_1 + F_2) =$$

$$\frac{h_x h_y}{4}(f(x_{0,}y_0) + f(x_{0,}y_2) + f(x_{2,}y_0) + f(x_{2,}y_2) + 2[f(x_{0,}y_0) + f(x_{1,}y_0) + f(x_{0,}y_1) + f(x_{1,}y_1)] + 4f(x_{1,}y_1))$$